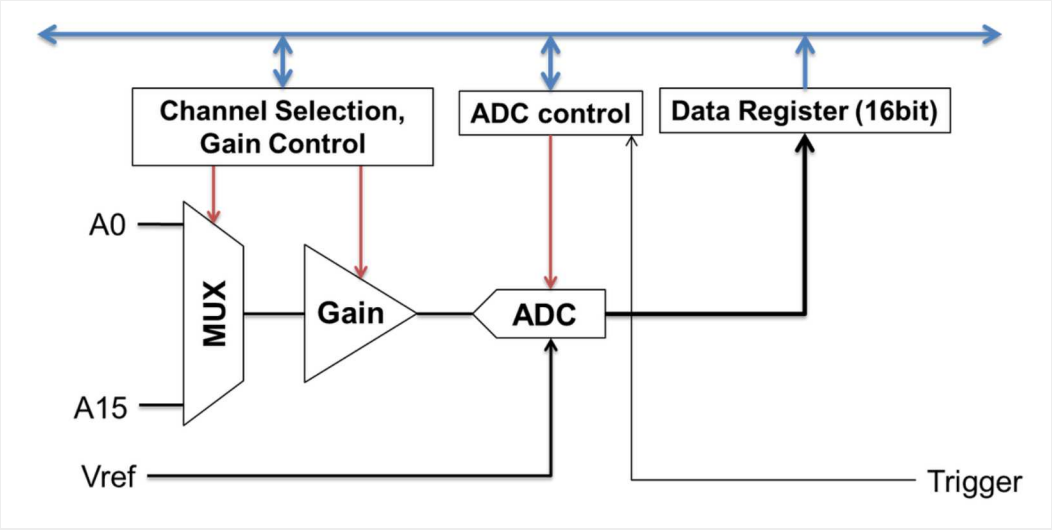
Benn Thomsen

Embedding Engineering Design

Analogue Input

The AVR Atmega 2560 contains a single 10 bit analogue to digital converter (ADC) with a maximum sample rate of 15kS/s at full resolution. It uses a successive approximation type ADC. The AVR Atmega 2560 provides 16 analogue input pins that are selected using the MUX that is before the ADC. The input range is from 0-Vcc, where Vcc is typically 5V. Differential voltage measurements can also be made using four independent differential channels, or 14 channels referenced to two reference voltages. A simplified schematic of the ADC subsystems is shown in figure 1.



— Figure 1. Simplified ADC schematic showing the main parts and registers

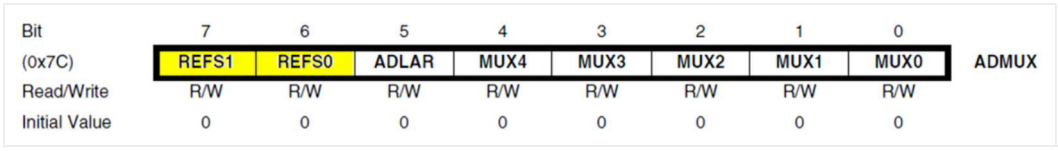
The ADC can be operated in the following three modes

1. **Single conversion**: Started by writing a logical one to the ADC Start Conversion bit
2. **Triggered conversion**: The conversion is initiated by the rising edge of the trigger input
3. **Free running**:The next conversion takes place immediately after the previous conversion is finished

Before using the ADC it is necessary to select the reference voltage source and the ADC clock frequency. The voltage reference source options are shown in table 1 and are determined by setting bits 6,7 in the ADCMUX register shown in Figure 2.

REFS1	REFS0	Voltage Reference Selection[1]
0	0	AREF, Internal V_{REF} turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Internal 1.1V Voltage Reference with external capacitor at AREF pin
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

— Table 1. ADC reference voltage source

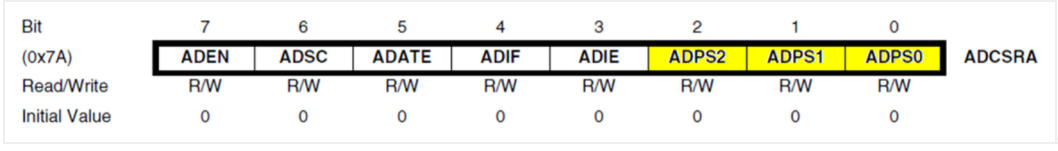


— Figure 2. ADC multiplexer register, showing the reference source bits

The ADC pre-scaler controls the internal ADC clock that controls the conversion process. By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be as high as 1000kHz to get a higher sample rate. The conversion process requires 13-14 ADC clock cycles and this sets an upper limit on the sampling frequency. The pre-scaler options are shown in table 2.

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

— Table 2. ADC pre-scaler values



— Figure 3. ADC Control and Status Register, with clock pre-scaler bits highlighted

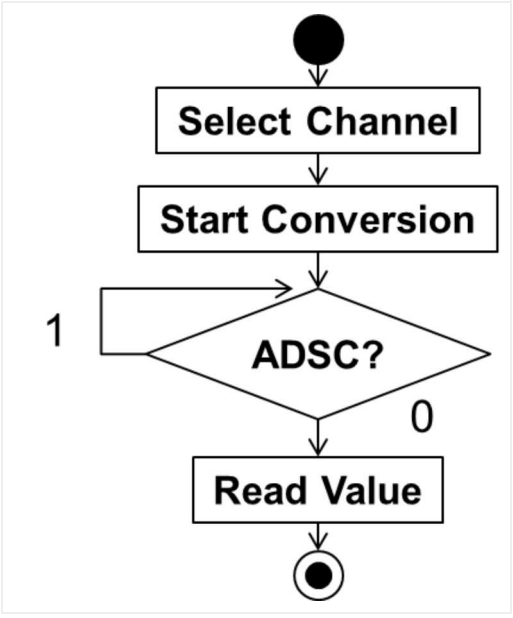
When using a 16MHz oscillator then using a prescale value of 128 will give ADC clock frequency of 125kHz which is well within the 50kHz and 200kHz limits. It is also necessary to turn the ADC on and to ensure that the ADC is fully initialised carryout a single conversion by setting the ADSC bit in the ADCSRA register. Note this first conversion takes 25 clock cycles. The following function can be used to initialise the ADC for single conversion operation.

```
1  void adc_init(void){
2
3  //16MHz/128 = 125kHz the ADC reference clock
4
5  ADCSRA |= ((1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0));
6
7  ADMUX |= (1<<REFS0);        //Set Voltage reference to Avcc (5v)
8
9  ADCSRA |= (1<<ADEN);        //Turn on ADC
10
11 ADCSRA |= (1<<ADSC); }      //Do an initial conversion
```

Bit	7	6	5	4	3	2	1	0	
(0x7B)	–	ACME	–	–	MUX5	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

— Figure 4. ADC Control and status register B

Once the ADC is initialised single conversions on any channel are carried out by first selecting the desired channel, starting the conversion and then polling the conversion complete flag (ADSC bit in the ADCSRA register, shown in figure 3) until the conversion is complete (ADSC=1 whilst the conversion is in progress and is set to ADSC=0 when the conversion is complete), then finally reading the converted value as shown in the UML activity chart of figure 5.



— Figure 5. ADC single conversion UML activity chart

The following function can be used to acquire a single conversion from the specified single ended input channel

```
1  uint16_t read_adc(uint8_t channel){
2  ADMUX &= 0xE0;           //Clear bits MUX0-4
3  ADMUX |= channel&0x07;   //Defines the new ADC channel to be read by
4  ADCSRB = channel&(1<<3); //Set MUX5
5  ADCSRA |= (1<<ADSC);      //Starts a new conversion
6  while(ADCSRA & (1<<ADSC));  //Wait until the conversion is done
7  return ADCW;}         //Returns the ADC value of the chosen channel
```

This example allows you to select any of the 16 input channels that are available on the Atmega2560 for single ended voltage conversion, that is the voltage on the input pin is measured with respect the the microprocessor ground. The inputs can also be configured for differential voltage measurement by choosing the correct setting of the MUX bits. For more information look at table 26.4 in section 26.8.2 of the datasheet (pg. 290).